**C++ Programming Questions and Answers – Error Handling**

1. Which keyword is used to handle the expection?  
a) try  
b) throw  
c) catch  
d) handler

Answer: c  
Explanation: When we found a exception in the program, We need to throw that and we handle that by using the catch keyword.

2. Which is used to throw a exception?  
a) throw  
b) try  
c) catch  
d) handler

Answer: a  
Explanation: throw keyword is used to throw an exception.  
eg:

if(divisor == 0)

{

throw "Divide by zero error";

}

3. What is the use of the ‘finally’ keyword?  
a) It used to execute at the starting of the program  
b) It will be executed at the end of the program even if the exception arised  
c) It will be executed at the starting of the program even if the exception arised  
d) It will be executed at the middle of the program even if the exception arised

Answer: b  
Explanation: finally keyword will be executed at the end of all the exception.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. double division(int a, int b)
4. {
5. if (b == 0)
6. {
7. throw "Division by zero condition!";
8. }
9. return (a / b);
10. }
11. int main ()
12. {
13. int x = 50;
14. int y = 0;
15. double z = 0;
16. try
17. {
18. z = division(x, y);
19. cout << z << endl;
20. }
21. catch (const char\* msg)
22. {
23. cerr << msg << endl;
24. }
25. return 0;
26. }

a) 50  
b) 0  
c) Division by zero condition!  
d) Error

Answer: c  
Explanation: It’s a mathematical certainty, We can’t divide by zero, So we’re arising a exception.  
Output:

$ g++ excep.cpp

$ a.out

Division by zero condition!

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main ()
4. {
5. try
6. {
7. throw 20;
8. }
9. catch (int e)
10. {
11. cout << "An exception occurred " << e << endl;
12. }
13. return 0;
14. }

a) 20  
b) An exception occurred  
c) Error  
d) An exception occurred 20

Answer: d  
Explanation: We are handling the exception by throwing that number. So the output is printed with the given number.  
Output:

$ g++ excep1.cpp

$ a.out

An exception occurred 20

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. class myexception: public exception
5. {
6. virtual const char\* what() const throw()
7. {
8. return "My exception";
9. }
10. } myex;
11. int main ()
12. {
13. try
14. {
15. throw myex;
16. }
17. catch (exception& e)
18. {
19. cout << e.what() << endl;
20. }
21. return 0;
22. }

a) Exception  
b) Error  
c) My exception  
d) Runtime error

Answer: c  
Explanation: This is a standard exception handler used in the class.  
Output:

$ g++ excep2.cpp

$ a.out

My exception

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. int main ()
5. {
6. try
7. {
8. int\* myarray = new int[1000];
9. cout << "allocated";
10. }
11. catch (exception& e)
12. {
13. cout << "Standard exception: " << e.what() << endl;
14. }
15. return 0;
16. }

a) Allocated  
b) Standard exception  
c) Depends on the memory  
d) Error

Answer: c  
Explanation: In this program, We are allocating the memory for array. If it is allocated means, no exception will arise and if there is no size in memory means, Exception will arise.  
Output:

$ g++ excep3.cpp

$ a.out

allocated

8. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. struct MyException : public exception
5. {
6. const char \* what () const throw ()
7. {
8. return "C++ Exception";
9. }
10. };
11. int main()
12. {
13. try
14. {
15. throw MyException();
16. }
17. catch(MyException& e)
18. {
19. cout << "Exception caught" << std::endl;
20. cout << e.what() << std::endl;
21. }
22. catch(std::exception& e)
23. {
24. }
25. }

a) C++ Exception  
b) Exception caught  
c) Exception caught

C++ Exception

d) Error

Answer: c  
Explanation: We are defining the user-defined exception in this program.  
Output:

$ g++ excep4.cpp

$ a.out

C++ Exception

Exception caught

9. How do define the user-defined exceptions?  
a) inheriting and overriding exception class functionality  
b) overriding class functionality  
c) inheriting class functionality  
d) delting and adding class member

Answer: a  
Explanation: User defined exceptions can be done by inheriting and overriding the exception class functionality.

10. Which exception is thrown by dynamic\_cast?  
a) bad\_cast  
b) bad\_typeid  
c) bad\_exception  
d) bad\_alloc

Answer: a  
Explanation: bad\_cast exception is thrown by dynamic\_cast.

# C++ Programming Questions and Answers – Grouping of Exceptions

1. How many types of exception handling are there in c++?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: b  
Explanation: There are two types of exception handling in c++. They are synchronous exception handling and asynchronous exception handling.

2. How many runtime error messages associated with exception?  
a) 2  
b) 4  
c) 5  
d) infinite

Answer: b  
Explanation: There are four runtime error messages associated with exceptions. They are overflow\_error, range\_error, system\_error and underflow\_error.

3. Which block should be placed after try block?  
a) catch  
b) throw  
c) either catch or throw  
d) handler  
View Answer

Answer: a  
Explanation: Syntax of try catch block:

try{

// do something

}

catch{

// catch respective error.

}

finally{

// do something after trying or catching error i.e. run this block in both cases.

}

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. double a = 10, b = 5, res;
6. char Operator = '/';
7. try
8. {
9. if (b == 0)
10. throw "Division by zero not allowed";
11. res = a / b;
12. cout << a << " / " << b << " = " << res;
13. }
14. catch(const char\* Str)
15. {
16. cout << "**\n** Bad Operator: " << Str;
17. }
18. return 0;
19. }

a) 10  
b) 2  
c) Bad Operator  
d) 10 / 5 = 2

Answer: d  
Explanation: In this program, We are dividing the two variables and printing the result. If any one of the operator is zero means, it will arise a exception.  
Output:

$ g++ gex.cpp

$ a.out

10 / 5 =2

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. try
6. {
7. throw 1;
8. }
9. catch (int a)
10. {
11. cout << "exception number: " << a << endl;
12. return 0;
13. }
14. cout << "No exception " << endl;
15. return 0;
16. }

a) No exception  
b) exception number  
c) exception number: 1  
d) exception number: 5

Answer: c  
Explanation: If we caught a integer value means, there will be an exception, if it is not a integer, there will not be a exception.  
Output:

$ g++ gex1.cpp

$ a.out

exception number: 1

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int a = 10, b = 20, c = 30;
6. float d;
7. try
8. {
9. if ((a - b) != 0)
10. {
11. d = c / (a - b);
12. cout << d;
13. }
14. else
15. {
16. throw(a - b);
17. }
18. }
19. catch (int i)
20. {
21. cout<<"Answer is infinite "<<i;
22. }
23. }

a) 10  
b) -3  
c) 15  
d) 17

Answer: b  
Explanation: We are manipulating the values, if there is any infinite value means, it will raise an exception.  
Output:

$ g++ gex2.cpp

$ a.out

-3

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. void test(int x)
4. {
5. try
6. {
7. if (x > 0)
8. throw x;
9. else
10. throw 'x';
11. }
12. catch(int x)
13. {
14. cout<<"integer:"<<x;
15. }
16. catch(char x)
17. {
18. cout << "character:" << x;
19. }
20. }
21. int main()
22. {
23. test(10);
24. test(0);
25. }

a) integer:10character:x  
b) integer:10  
c) character:x  
d) character:10

Answer: a  
Explanation: We are passing the integer and character and catching it by using multiple catch statement.  
Output:

$ g++ gex3.cpp

$ a.out

integer:10character:x

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. void PrintSequence(int StopNum)
4. {
5. int Num;
6. Num = 1;
7. while (true)
8. {
9. if (Num >= StopNum)
10. throw Num;
11. cout << Num << endl;
12. Num++;
13. }
14. }
15. int main(void)
16. {
17. try
18. {
19. PrintSequence(2);
20. }
21. catch(int ExNum)
22. {
23. cout << "exception: " << ExNum << endl;
24. }
25. return 0;
26. }

a) 1  
b) exception: 2  
c) 1

exception: 2

d) exception: 4

Answer: c  
Explanation: In this program, We are printing one and raising a exception at 2.  
Output:

$ g++ gex4.cpp

$ a.out

1

exception: 2

9. Pick out the correct answer.  
a) Exceptions are not suitable for critical points in code  
b) Exception are suitable for critical points in code  
c) Exceptions are used when postconditions of a function cannot be satisfied  
d) Throw block should be placed after try block

Answer: a  
Explanation: If there is many number of exceptions in the program means, We have to use multiple catch statement and it is hard to keep track of the program.

10. When exceptions are used?  
a) To preserve the program  
b) Exceptions are used when postconditions of a function cannot be satisfied  
c) Exceptions are used when postconditions of a function can be satisfied  
d) Exceptions are used when preconditions of a function cannot be satisfied

Answer: c  
Explanation: Exceptions are used when postconditions of a function can be satisfied.

# C++ Programming Questions and Answers – Catching Exceptions

1. How many parameters does the throw expression can have?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: a  
Explanation: In c++ program, We can be able to throw only one error at a time.

2. Where exception are handled?  
a) inside the program  
b) outside the regular code  
c) both inside or outside  
d) main program

Answer: b  
Explanation: Exception are handled outside the regular code.

3. Which is used to check the error in the block?  
a) try  
b) throw  
c) catch  
d) handler

Answer: a  
Explanation: The try block is used to check for errors, if there is any error means, it can throw it to catch block.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. class myexception: public exception
5. {
6. virtual const char\* what() const throw()
7. {
8. return "exception arised";
9. }
10. } myex;
11. int main ()
12. {
13. try
14. {
15. throw myex;
16. }
17. catch (exception& e)
18. {
19. cout << e.what() << endl;
20. }
21. return 0;
22. }

a) exception arised  
b) error  
c) exception  
d) runtime error

Answer: a  
Explanation: In this program, We are arising a standard exception and catching that and returning a statement.  
Output:

$ g++ goe.cpp

$ a.out

exception arised

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int age=5;
6. try
7. {
8. if (age < 0)
9. throw "Positive Number Required";
10. cout << age << "**\n\n**";
11. }
12. catch(const char\* Message)
13. {
14. cout << "Error: " << Message;
15. }
16. return 0;
17. }

a) 5  
b) 10  
c) 15  
d) Positive Number Required

Answer: a  
Explanation: In this program, We are checking the age of a person, If it is zero means, We will arise a exception.  
Output:

$ g++ goe1.cpp

$ a.out

5

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. double division(int a, int b)
4. {
5. if ( b == 0 )
6. {
7. throw "Division by zero condition!";
8. }
9. return (a / b);
10. }
11. int main ()
12. {
13. int x = 50;
14. int y = 0;
15. double z = 0;
16. try
17. {
18. z = division(x, y);
19. cout << z << endl;
20. }
21. catch (const char\* msg)
22. {
23. cout << msg << endl;
24. }
25. return 0;
26. }

a) 50  
b) 0  
c) Division by zero condition!  
d) 100

Answer: c  
Explanation: We are dividing the values and if one of the values is zero means, We are arising an exception.  
Output:

$ g++ goe2.cpp

$ a.out

Division by zero condition!

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. int main()
5. {
6. double Op1 = 10, Op2 = 5, Res;
7. char Op;
8. try
9. {
10. if (Op != '+' && Op != '-' && Op != '\*' && Op != '/')
11. throw Op;
12. switch(Op)
13. {
14. case '+':
15. Res = Op1 + Op2;
16. break;
17. case '-':
18. Res = Op1 - Op2;
19. break;
20. case '\*':
21. Res = Op1 \* Op2;
22. break;
23. case '/':
24. Res = Op1 / Op2;
25. break;
26. }
27. cout << "**\n**" << Op1 << " " << Op << " "<< Op2 << " = " << Res;
28. }
29. catch (const char n)
30. {
31. cout << n << " is not a valid operator";
32. }
33. return 0;
34. }

a) 15  
b) 5  
c) 2  
d) is not a valid operator

Answer: d  
Explanation: It will arise a exception because we missed a operator.  
Output:

$ g++ goe3.cpp

$ a.out

is not a valid operator

8. What will be the output of the following C++ code?

1. #include<iostream>
2. #include "math.h"
3. using namespace std;
4. double MySqrt(double d)
5. {
6. if (d < 0.0)
7. throw "Cannot take sqrt of negative number";
8. return sqrt(d);
9. }
10. int main()
11. {
12. double d = 5;
13. cout << MySqrt(d) << endl;
14. }

a) 5  
b) 2.236  
c) Error  
d) Cannot take sqrt of negative number

Answer: b  
Explanation: We are finding the square root of the number, if it is a positive number, it can manipulate, Otherwise it will arise a exception.  
Output:

$ g++ goe4.cpp

$ a.out

2.236

9. How to handle the exception in constructor?  
a) We have to throw an exception  
b) We have to return the exception  
c) We have to throw an exception & return the exception  
d) We have to catch an exception

Answer: a  
Explanation: As a constructor don’t have a return type, We have to throw the exception.

10. What should present when throwing a object?  
a) constructor  
b) copy-constructor  
c) destructor  
d) copy-destructor

Answer: b  
Explanation: copy-constructor is mandatory for throwing a object.

# C++ Programming Questions and Answers – Resource Management

1. What can go wrong in resource management on c++?  
a) Leakage  
b) Exhaustion  
c) Dangling  
d) Exception

Answer: d  
Explanation: If there is any mishap in memory or resource management means, the problems that are mentioned above can happen.

2. When do we call that resource is leaked?  
a) Arise of compile time error  
b) It cannot be accessed by any standard mean  
c) Arise of runtime error  
d) It can be accessed by any standard mean

Answer: b  
Explanation: Resource is said to be leaked when it cannot be accessed by any means of standard mean.

3. What kind of error can arise when there is a problem with memory?  
a) Segmentation fault  
b) Produce an error  
c) Both Segmentation fault & Produce an error  
d) runtime error

Answer: a  
Explanation: segmentation fault error can arise when there is a problem with memory.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <new>
3. using namespace std;
4. int main ()
5. {
6. int i, n;
7. int \* p;
8. i = 2;
9. p= new (nothrow) int[i];
10. if (p == 0)
11. cout << "Error: memory could not be allocated";
12. else
13. {
14. for (n=0; n<i; n++)
15. {
16. p[n] = 5;
17. }
18. for (n = 0; n < i; n++)
19. cout << p[n];
20. delete[] p;
21. }
22. return 0;
23. }

a) 5  
b) 55  
c) 555  
d) Error: memory could not be allocated

Answer: b  
Explanation: As we had given i value as 2, It will print the 5 for two times.  
Output:

$ g++ res.cpp

$ a.out

55

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main(void)
4. {
5. const char \*one = "Test";
6. cout << one << endl;
7. const char \*two = one;
8. cout << two << endl;
9. return 0;
10. }

a) Test  
b) TestTest  
c) Te  
d) TestTe

Answer: b  
Explanation: We are copying the values from one variable to other, So it is printing is TestTest  
Output:

$ g++ res1.cpp

$ a.out

TestTest

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int funcstatic(int)
4. {
5. int sum = 0;
6. sum = sum + 10;
7. return sum;
8. }
9. int main(void)
10. {
11. int r = 5, s;
12. s = funcstatic(r);
13. cout << s << endl;
14. return 0;
15. }

a) 10  
b) 15  
c) error  
d) 20

Answer: a  
Explanation: Even Though we passed the value, we didn’t caught to manipulate it, So it is printing as 10.  
Output:

$ g++ res2.cpp

$ a.out

10

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include<string.h>
3. using namespace std;
4. int main()
5. {
6. try
7. {
8. char \*p;
9. strcpy(p, "How r u");
10. }
11. catch(const exception& er)
12. {
13. }
14. }

a) How r u  
b) segmentation fault  
c) error  
d) runtime error

Answer: b  
Explanation: As we are using a pointer value to copy a string, So it will be producing a runtime error.  
Output:

$ g++ res3.cpp

$ a.out

segmentation fault

8. What is meant by garbage collection?  
a) The form of manual memory management  
b) The form of automatic memory management  
c) Used to replace the variables  
d) Used to delete the variables

Answer: b  
Explanation: The garbage collection attempts to reclaim memory occupied by objects that are no longer in use by the program.

9. What are the operators available in C++ for dynamic allocation and de-allocation of memories?  
a) new  
b) delete  
c) compare  
d) both new & delete

Answer: d  
Explanation: new and delete operators are mainly used to allocate and deallocate during runtime.

10. Which is used to solve the memory management problem in c++?  
a) smart pointers  
b) arrays  
c) stack  
d) queue

Answer: a  
Explanation: In C++, smart pointers are used to manage memory issues like deallocate memory after use, checking bounds, etc.

# C++ Programming Questions and Answers – Exceptions That Are Not Errors

1. Which is used to handle the exceptions in c++?  
a) catch handler  
b) handler  
c) exception handler  
d) throw

Answer: c  
Explanation: Exception handler is used to handle the exceptions in c++.

2. Which type of program is recommended to include in try block?  
a) static memory allocation  
b) dynamic memory allocation  
c) const reference  
d) pointer

Answer: b  
Explanation: While during dynamic memory allocation, Your system may not have sufficient resources to handle it, So it is better to use it inside the try block.

3. Which statement is used to catch all types of exceptions?  
a) catch()  
b) catch(Test t)  
c) catch(…)  
d) catch(Test)

Answer: c  
Explanation: This catch statement will catch all types of exceptions that arises in the program.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int x = -1;
6. try
7. {
8. if (x < 0)
9. {
10. throw x;
11. }
12. else
13. {
14. cout<<x;
15. }
16. }
17. catch (int x )
18. {
19. cout << "Exception occurred: Thrown value is " << x << endl;
20. }
21. return 0;
22. }

a) -1  
b) 0  
c) Exception occurred: Thrown value is -1  
d) Error

Answer: c  
Explanation: As the given value is -1 and according to the condition, We are arising an exception.  
Output:

$ g++ etae.cpp

$ a.out

Exception occurred: Thrown value is -1

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <typeinfo>
3. using namespace std;
4. class Polymorphic {virtual void Member(){}};
5. int main ()
6. {
7. try
8. {
9. Polymorphic \* pb = 0;
10. typeid(\*pb);
11. }
12. catch (exception& e)
13. {
14. cerr << "exception caught: " << e.what() << endl;
15. }
16. return 0;
17. }

a) exception caught: std::bad\_typeid  
b) exception caught: std::bad\_alloc  
c) exception caught: std::bad\_cast  
d) exception caught: std::bad\_id

Answer: a  
Explanation: In this program, We used a bad type id for the polymorphic operator, So it is arising an bad\_typeid exception.  
Output:

$ g++ etae.cpp

$ a.out

exception caught: std::bad\_typeid

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. void myunexpected ()
5. {
6. cout << "unexpected handler called**\n**";
7. throw;
8. }
9. void myfunction () throw (int,bad\_exception)
10. {
11. throw 'x';
12. }
13. int main (void)
14. {
15. set\_unexpected (myunexpected);
16. try
17. {
18. myfunction();
19. }
20. catch (int)
21. {
22. cout << "caught int**\n**";
23. }
24. catch (bad\_exception be)
25. {
26. cout << "caught bad\_exception**\n**";
27. }
28. catch (...)
29. {
30. cout << "caught other exception **\n**";
31. }
32. return 0;
33. }

a) unexpected handler called  
b) caught bad\_exception  
c) caught other exception  
d) both unexpected handler called & caught bad\_exception

Answer: d  
Explanation: In this program, We are calling set\_unexpected and myfunction, So it is printing the output as the given.  
Output:

$ g++ etae.cpp

$ a.out

unexpected handler called

caught bad\_exception

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int x = -1;
6. char \*ptr;
7. ptr = new char[256];
8. try
9. {
10. if (x < 0)
11. {
12. throw x;
13. }
14. if (ptr == NULL)
15. {
16. throw " ptr is NULL ";
17. }
18. }
19. catch (...)
20. {
21. cout << "Exception occurred: exiting "<< endl;
22. }
23. return 0;
24. }

a) -1  
b) ptr is NULL  
c) exception occured: exiting  
d) 1

Answer: c  
Explanation: catch(…) is used to catch all types of exceptions arising in the program.  
Output:

$ g++ etea.cpp

$ a.out

Exception occured: exiting

8. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. void myunexpected ()
5. {
6. cout << "unexpected called**\n**";
7. throw 0;
8. }
9. void myfunction () throw (int)
10. {
11. throw 'x';
12. }
13. int main ()
14. {
15. set\_unexpected (myunexpected);
16. try
17. {
18. myfunction();
19. }
20. catch (int)
21. {
22. cout << "caught int**\n**";
23. }
24. catch (...)
25. {
26. cout << "caught other exception**\n**";
27. }
28. return 0;
29. }

a) caught other exception  
b) caught int  
c) unexpected called  
d) both caught int & unexpected called

Answer: d  
Explanation: As we are calling set\_unexpected (myunexpected) function, this is printing as unexpected called and because of operator compliance it is arising an exception.  
Output:

$ g++ etea.cpp

$ a.out

unexpected called

caught int

9. How to handle error in the destructor?  
a) throwing  
b) terminate  
c) both throwing & terminate  
d) try

Answer: b  
Explanation: It will not throw an exception from the destructor but it will the process by using terminate() function.

10. What kind of exceptions are available in c++?  
a) handled  
b) unhandled  
c) static  
d) dynamic

Answer: b  
Explanation: unhandled kind of exceptions are available in c++.

# C++ Programming Questions and Answers – Exception Specifications

1. What is meant by exception specification?  
a) A function is limited to throwing only a specified list of exceptions  
b) A catch can catch all types of exceptions  
c) A function can throw any type of exceptions  
d) A try can catch all types of exceptions

Answer: a  
Explanation: C++ provides a mechanism to ensure that a given function is limited to throwing only a specified list of exceptions. It is called an exception specification.

2. Identify the correct statement about throw(type).  
a) A function can throw any type of exceptions  
b) A function can throw an exception of certain type only  
c) A function can’t throw any type of exception  
d) A function can catch all types of exceptions

Answer: b  
Explanation: A function can throw an exception of certain type only.

3. What will happen when a programs throws any other type of exception other than specified?  
a) terminate  
b) arise an error  
c) run  
d) throw

Answer: b  
Explanation: Because there is no way defined to catch that exception and as we know if an exception is not caught then error arises.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. void empty() throw()
4. {
5. cout << "In empty()";
6. }
7. void with\_type() throw(int)
8. {
9. cout << "Will throw an int";
10. throw(1);
11. }
12. int main()
13. {
14. try
15. {
16. empty();
17. with\_type();
18. }
19. catch (int)
20. {
21. cout << "Caught an int";
22. }
23. }

a) In empty()  
b) Will throw an int  
c) Caught an int  
d) All of the mentioned

Answer: d  
Explanation: It will print all three because we are calling all functions in the main().  
Output:

$ g++ exs.cpp

$ a.out

In empty()Will throw an intCaught an int

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. #include <typeinfo>
4. using namespace std;
5. class Test1
6. {
7. virtual int Funct()
8. {
9. }
10. };
11. int main ()
12. {
13. try
14. {
15. Test1 \* var = NULL;
16. typeid (\*var);
17. }
18. catch (std::exception& typevar)
19. {
20. cout << "Exception: " << typevar.what() << endl;
21. }
22. return 0;
23. }

a) NULL  
b) Exception:bad\_alloc  
c) Exception:std:bad\_typeid  
d) Exception:std:bad\_type

Answer: c  
Explanation: As we are using a bad type on pointers, So it is arising an error.  
Output:

$ g++ exs1.cpp

$ a.out

Exception:std:bad\_typeid

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <string>
3. #include<typeinfo>
4. using namespace std;
5. int main( )
6. {
7. try
8. {
9. string strg1("Test");
10. string strg2("ing");
11. strg1.append(strg2, 4, 2);
12. cout << strg1 << endl;
13. }
14. catch (exception &e)
15. {
16. cout << "Caught: " << e.what() << endl;
17. cout << "Type: " << typeid(e).name() << endl;
18. };
19. return 0;
20. }

a) out of range  
b) bad type\_id  
c) bad allocation  
d) bad typ

Answer: a  
Explanation: As we are using out of bound value on strings, So it arising an exception.  
Output:

$ g++ exs2.cpp

$ a.out

Caught: basic\_string::append

Type: St12out\_of\_range

#include <string>

7. What will be the output of the following C++ code?

1. #include <typeinfo>
2. #include <iostream>
3. using namespace std;
4. class Myshape
5. {
6. public:
7. virtual void myvirtualfunc() const {}
8. };
9. class mytriangle: public Myshape
10. {
11. public:
12. virtual void myvirtualfunc() const
13. {
14. };
15. };
16. int main()
17. {
18. Myshape Myshape\_instance;
19. Myshape &ref\_Myshape = Myshape\_instance;
20. try
21. {
22. mytriangle &ref\_mytriangle = dynamic\_cast<mytriangle&>(ref\_Myshape);
23. }
24. catch (bad\_cast)
25. {
26. cout << "Can't do the dynamic\_cast lor!!!" << endl;
27. cout << "Caught: bad\_cast exception. Myshape is not mytriangle.**\n**";
28. }
29. return 0;
30. }

a) Can’t do the dynamic\_cast lor!!!  
b) Caught: bad\_cast exception. Myshape is not mytriangle.  
c) Can’t able to create the dynamic instance for the triangle, So it is arising an exception  
d) Myshape is not mytriangle

Answer: c  
Explanation: As we can’t able to create the dynamic instance for the triangle, So it is arising an exception.  
Output:

$ g++ exs3.cpp

$ a.out

Can't do the dynamic\_cast lor!!!

Caught: bad\_cast exception. Myshape is not mytriangle.

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. char\* ptr;
6. unsigned long int Test = sizeof(size\_t(0) / 3);
7. cout << Test << endl;
8. try
9. {
10. ptr = new char[size\_t(0) / 3];
11. delete[ ] ptr;
12. }
13. catch (bad\_alloc &thebadallocation)
14. {
15. cout << thebadallocation.what() << endl;
16. };
17. return 0;
18. }

a) 4  
b) 2  
c) bad\_alloc  
d) depends on compiler

Answer: d  
Explanation: The size of unsigned long int always depends on compiler.  
Output:

$ g++ exs4.cpp

$ a.out

4

9. What do you mean by “No exception specification”?  
a) It throws nothing  
b) It can throw anything  
c) It can catch anything  
d) It can try anything

Answer: b  
Explanation: No exception specification that it can throw anything.

10. Which operations don’t throw anything?  
a) Operations which are reversible  
b) Operations which are irreversible  
c) Operations which are static  
d) Operations which are dynamic

Answer: b  
Explanation: Operations which are irreversible cannot throw anything.

# C++ Programming Questions and Answers – Uncaught Exceptions

1. What happens if try catch block is not used?  
a) arise an error  
b) program will run  
c) execute continuously  
d) wrong output

Answer: a  
Explanation: If try catch block is not used the exception thrown by the program will be uncaught hence will result into error(s).

2. Which handler is used to handle all types of exception?  
a) catch handler  
b) catch-all handler  
c) catch-none handler  
d) try-catch handler

Answer: b  
Explanation: To catch all types of exceptions, we use the catch-all handler.

3. Which operator is used as catch-all handler?  
a) ellipses operator  
b) ternary operator  
c) string operator  
d) unary operator

Answer: a  
Explanation: The ellipses operator can be represented as (…).

4. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5. protected:
6. int a;
7. public:
8. Base()
9. {
10. a = 34;
11. }
12. Base(int i)
13. {
14. a = i;
15. }
16. virtual ~Base()
17. {
18. if (a < 0) throw a;
19. }
20. virtual int getA()
21. {
22. if (a < 0)
23. {
24. throw a;
25. }
26. }
27. };
28. int main()
29. {
30. try
31. {
32. Base b(-25);
33. cout << endl << b.getA();
34. }
35. catch (int)
36. {
37. cout << endl << "Illegal initialization";
38. }
39. }

a) Illegal initialization  
b) Terminate called after throwing an instance of ‘int’  
c) Illegal initialization & terminate called after throwing an instance  
d) initialization

Answer: b  
Explanation: As we are throwing a negative number and we are using the only integer, So it is arising an error.  
Output:

$ g++ uce.cpp

$ a.out

terminate called after throwing an instance of 'int'

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. void terminator()
5. {
6. cout << "terminate" << endl;
7. }
8. void (\*old\_terminate)() = set\_terminate(terminator);
9. class Botch
10. {
11. public:
12. class Fruit {};
13. void f()
14. {
15. cout << "one" << endl;
16. throw Fruit();
17. }
18. ~Botch()
19. {
20. throw 'c';
21. }
22. };
23. int main()
24. {
25. try
26. {
27. Botch b;
28. b.f();
29. }
30. catch(...)
31. {
32. cout << "inside catch(...)" << endl;
33. }
34. }

a) one  
b) inside catch  
c)one

terminate

d)one

terminate

Aborted

Answer: d  
Explanation: This program uses set\_terminate as it is having an uncaught exception.  
Output:

$ g++ uce1.cpp

$ a.out

one

terminate

Aborted

6. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. #include <cstdlib>
4. using namespace std;
5. void myterminate ()
6. {
7. cerr << "terminate handler called";
8. abort();
9. }
10. int main (void)
11. {
12. set\_terminate (myterminate);
13. throw 0;
14. return 0;
15. }

a) terminate handler called  
b) aborted  
c) both terminate handler & Aborted  
d) runtime error

Answer: c  
Explanation: In this program, We are using set\_terminate to abort the program.  
Output:

$ g++ uce2.cpp

$ a.out

terminate handler called

Aborted

7. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. class Test1
4. {
5. };
6. class Test2 : public Test1 { };
7. void Funct();
8. int main()
9. {
10. try
11. {
12. Funct();
13. }
14. catch (const Test1&)
15. {
16. cerr << "Caught a exception" << endl;
17. }
18. return 0;
19. }
20. void Funct()
21. {
22. throw Test2();
23. }

a) Caught an exception  
b) NULL  
c) Both Caught an exception & NULL  
d) Caught a exception

Answer: a  
Explanation: In this program, We are arising with the exception by using the method in the class.  
Output:

$ g++ uce3.cpp

$ a.out

Caught a exception

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. #include <cstdlib>
4. #include <exception>
5. void Funct()
6. {
7. cout << "Funct() was called by terminate()." << endl;
8. exit(0);
9. }
10. int main()
11. {
12. try
13. {
14. set\_terminate(Funct);
15. throw "Out of memory!";
16. }
17. catch(int)
18. {
19. cout << "Integer exception raised." << endl;
20. }
21. return 0;
22. }

a) Integer exception raised  
b) Funct() was called by terminate()  
c) Integer exception not raised  
d) Integer exception raised.

Answer: b  
Explanation: As there is no integer in this program, We are printing Funct() was called by terminate().  
Output:

$ g++ uce4.cpp

$ a.out

Funct() was called by terminate().

9. What function will be called when we have an uncaught exception?  
a) catch  
b) throw  
c) terminate  
d) try

Answer: c  
Explanation: If we have an uncaught exception means, the compiler will throw the control of the program to terminate function.

10. What will not be called when the terminate() is raised in the constructor?  
a) main()  
b) class  
c) destructor  
d) constructor

Answer: c  
Explanation: To free the memory occupied by that object during initializing and destroy that object.

# C++ Programming Questions and Answers – Exceptions and Efficiency

1. What will happen when we move to try block far away from catch block?  
a) Reduces the amount of code in cache  
b) Increases the amount of code in cache  
c) Don’t alter anything  
d) Increases the amount of code

Answer: a  
Explanation: compilers may try to move the catch-code far away from the try-code, which reduces the amount of code to keep in cache normally, thus enhancing performance.

2. What will happen if an exception that is thrown may cause a whole load of objects to go out of scope?  
a) Terminate the program  
b) Produce a runtime error  
c) It will be added to the overhead  
d) Compilation error

Answer: c  
Explanation: It will be added to the overhead if an exception that is thrown may cause a whole load of objects to go out of scope.

3. What operation can be performed by destructor?  
a) Abort the program  
b) Resource cleanup  
c) Exit from the current block  
d) Terminate the program

Answer: b  
Explanation: It will be used to free all the resources that are used by the block of code during execution.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. int main ()
5. {
6. try
7. {
8. double\* i= new double[1000];
9. cout << "Memory allocated";
10. }
11. catch (exception& e)
12. {
13. cout << "Exception arised: " << e.what() << endl;
14. }
15. return 0;
16. }

a) Memory allocated  
b) Exception arised  
c) Depends on the computer memory  
d) Memory allocatedException arised

Answer: c  
Explanation: The value will be allocated, if there is enough memory in the system.  
Output:

$ g++ expef.cpp

$ a.out

Memory allocated

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. void test(int x)
4. {
5. try
6. {
7. if (x > 0)
8. throw x;
9. else
10. throw 'x';
11. }
12. catch(char)
13. {
14. cout << "Catch a integer and that integer is:" << x;
15. }
16. }
17. int main()
18. {
19. cout << "Testing multiple catches**\n**:";
20. test(10);
21. test(0);
22. }

a) Catch a integer and that integer is:10  
b) Error  
c) Runtime error  
d) Catch a integer and that integer is:25

Answer: c  
Explanation: As the catch is created with a wrong type, So it will  
arise a runtime error.  
Output:

$ g++ expef.cpp

$ a.out

Testing multiple catches

terminate called after throwing an instance of 'int'

:Aborted

6. What will be the output of the following C++ code?

1. #include <stdexcept>
2. #include <limits>
3. #include <iostream>
4. using namespace std;
5. void func(int c)
6. {
7. if (c < numeric\_limits<char> :: max())
8. throw invalid\_argument("MyFunc argument too large.");
9. else
10. {
11. cout<<"Executed";
12. }
13. }
14. int main()
15. {
16. try
17. {
18. func(256);
19. }
20. catch(invalid\_argument& e)
21. {
22. cerr << e.what() << endl;
23. return -1;
24. }
25. return 0;
26. }

a) Invalid arguments  
b) Executed  
c) Error  
d) Runtime error

Answer: b  
Explanation: As we are throwing the function and catching it with a correct data type, So this program will execute.  
Output:

$ g++ expef.cpp

$ a.out

Executed

7. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. int main ()
5. {
6. int num = 3;
7. string str\_bad = "wrong number used";
8. try
9. {
10. if ( num == 1 )
11. {
12. throw 5;
13. }
14. if ( num == 2 )
15. {
16. throw 1.1f;
17. }
18. if ( num != 1 || num != 2 )
19. {
20. throw str\_bad;
21. }
22. }
23. catch (int a)
24. {
25. cout << "Exception is: " << a << endl;
26. }
27. catch (float b)
28. {
29. cout << "Exception is: " << b << endl;
30. }
31. catch (...)
32. {
33. cout << str\_bad << endl;
34. }
35. return 0;
36. }

a) Exception is 5  
b) Exception is 1.1f  
c) Wrong number used  
d) Exception is 1.6g

Answer: c  
Explanation: As we are giving 3 to num, It is arising an exception named  
“wrong number used”.  
Output:

$ g++ expef.cpp

$ a.out

wrong number used

8. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. double division(int a, int b)
4. {
5. if (b == 0)
6. {
7. throw "Division by zero condition!";
8. }
9. return (a / b);
10. }
11. int main ()
12. {
13. int x = 50;
14. int y = 0;
15. double z = 0;
16. try
17. {
18. z = division(x, y);
19. cout << z << endl;
20. }
21. catch (const msg)
22. {
23. cerr << msg << endl;
24. }
25. return 0;
26. }

a) 50  
b) 0  
c) Division by zero condition  
d) Error

Answer: d  
Explanation: As we missed the data type in the catch block, It will arise an error.

9. What is the main purpose of the constructor?  
a) Begin the execution of class  
b) Include the macros for the program  
c) Establish the class invariant  
d) Terminate the program

Answer: c  
Explanation: The purpose of a constructor is to establish the class invariant. To do that, it often needs to acquire system resources or in general perform an operation that may fail.

10. Why is it expensive to use objects for the exception?  
a) Exception object is created only if an error actually happens  
b) Because of execution time  
c) Memory space involved in creating an exception object  
d) Because of time and space

Answer: a  
Explanation: If an error occurs in the program, then only exception object is created otherwise, It will not be created. since throwing an exception triggers a bunch of actions during the stack unrolling, like invoking the the destructor of all the objects that has been created up to the point in which we are able to catch the exception, and invoking the destructor methods can imply flushing streams and freeing memory which can be expensive as well. Therefore, it’s expensive to use in the program.

# C++ Programming MCQ – Exception Handling

1. What is an exception in C++ program?  
a) A problem that arises during the execution of a program  
b) A problem that arises during compilation  
c) Also known as the syntax error  
d) Also known as semantic error

Answer: a  
Explanation: An exception is defined as the problem in C++ program that arises during the execution of the program for example divide by zero error.

2. By default, what a program does when it detects an exception?  
a) Continue running  
b) Results in the termination of the program  
c) Calls other functions of the program  
d) Removes the exception and tells the programmer about an exception

Answer: b  
Explanation: By default, whenever a program detects an exception the program crashes as it does not know how to handle it hence results in the termination of the program.

3. Why do we need to handle exceptions?  
a) To avoid unexpected behaviour of a program during run-time  
b) To let compiler remove all exceptions by itself  
c) To successfully compile the program  
d) To get correct output

Answer: a  
Explanation: We need to handle exceptions in a program to avoid any unexpected behaviour during run-time because that behaviour may affect other parts of the program. Also, an exception is detected during run-time, therefore, a program may compile successfully even with some exceptions cases in your program.

4. How Exception handling is implemented in the C++ program?  
a) Using Exception keyword  
b) Using try-catch block  
c) Using Exception block  
d) Using Error handling schedules

Answer: b  
Explanation: C++ provides a try-catch block to handle exceptions in your program.

5. What is the correct syntax of the try-catch block?  
a)try

{

// programable codes.....

}

catch(Exceptions)

{

// Code for handling exceptions....

}

b)try()

{

// programable codes.....

}

catch(Exceptions)

{

// Code for handling exceptions....

}

c)try

{

// programable codes.....

}

catch

{

// Code for handling exceptions....

}

d)try()

{

// programable codes.....

}

catch

{

// Code for handling exceptions....

}

Answer: a  
Explanation: Try-catch block has the following syntax:

try{

// codes that needs to check for exceptions

}

catch(Exception E1){

// codes for handling exception....

// Exception E denotes the type of exception this block is handling.

}

catch(Exception E2){

// other exception that needs to be handled...

}

6. Which part of the try-catch block is always fully executed?  
a) try part  
b) catch part  
c) finally part  
d) throw part

Answer: c  
Explanation: finally part of the try-catch block is always executed whether exceptions are caught or not.

7. Which of the following is an exception in C++?  
a) Divide by zero  
b) Semicolon not written  
c) Variable not declared  
d) An expression is wrongly written

Answer: a  
Explanation: Exceptions are those which are encountered during run-time of the program. semicolon, variable not declared and the wrong expression are compile-time errors, therefore, they are not exceptions. Divide by zero is the problem that is encountered during run-time, therefore, it is an exception.

8. What is an error in C++?  
a) Violation of syntactic and semantic rules of a languages  
b) Missing of Semicolon  
c) Missing of double quotes  
d) Violation of program interface

Answer: a  
Explanation: An error occurs when rules and laws of a language is violated while writing programs in that language.

9. What is the difference between error and exception?  
a) Both are the same  
b) Errors can be handled at the run-time but the exceptions cannot  
c) Exceptions can be handled at the run-time but the errors cannot  
d) Both can be handled during run-time

Answer: c  
Explanation: Exceptions can be handled during run-time whereas errors cannot be because exceptions occur due to some unexpected conditions during run-time whereas about errors compiler is sure and tells about them during compile-time.

10. What are the different types of exceptions?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: b  
Explanation: There are two types of exceptions: Synchronous and asynchronous exceptions. Synchronous exceptions that are caused by the event which can be controlled by the program whereas Asynchronous exceptions are those which are beyond the control of the program.

11. Which keyword is used to throw an exception?  
a) try  
b) throw  
c) throws  
d) except

Answer: b  
Explanation: ‘throw’ keyword is used to throw exceptions if something bad happens.

12. What will be the output of the following C++ code?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

void func(int a, int b)

{

if(b == 0){

throw "This value of b will make the product zero. "

"So please provide positive values.**\n**";

}

else{

cout<<"Product of "<<a<<" and "<<b<<" is: "<<a\*b<<endl;

}

}

int main()

{

try{

func(5,0);

}

catch(const char\* e){

cout<<e;

}

}

a) 0  
b) 5  
c) This value of b will make the product zero. So please provide positive values.  
d) Product of 5 and 0 is: 0

Answer: c  
Explanation: As the value of b = 0 is provided to the func() and the function is throwing an exception whenever the value of b = 0. Therefore the function throws the exception which will be printed on the screen.  
Output:

$ ./a.out

This value of b will make the product zero. So please provide positive values.

13. What will be the output of the following C++ code?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

void func(int a, int b)

{

if(b == 0){

throw "This value of b will make the product zero. "

"So please provide positive values.**\n**";

}

else{

cout<<"Product of "<<a<<" and "<<b<<" is: "<<a\*b<<endl;

}

}

int main()

{

try{

func(5,0);

}

catch(char\* e){

cout<<e;

}

}

a) 0  
b) Aborted (core dumped)  
c) This value of b will make the product zero. So please provide positive values.  
d) Product of 5 and 0 is: 0

Answer: b  
Explanation: As the func() is throwing a const char\* string but we the catch block is not catching any const char\* exception i.e. exception thrown is not handled therefore the program results into Aborted(core dumped).  
Output:

$ ./a.out

terminate called after throwing an instance of 'char const\*'

Aborted (core dumped)

14. What is Re-throwing an exception means in C++?  
a) An exception that is thrown again as it is not handled by that catching block  
b) An exception that is caught twice  
c) An exception that is not handled in one caught hence thrown again  
d) All of the mentioned

Answer: d  
Explanation: Exception that is caught by a catch block but not handled by that catch block can be re-thrown by that catch block to further try-catch block.

15. What will be the output of the following C++ code?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

void func(int a, int b)

{

if(b < 1){

throw b;

}

else{

cout<<"Product of "<<a<<" and "<<b<<" is: "<<a\*b<<endl;

}

}

int main()

{

try

{

try

{

func(5,-1);

}

catch(int b)

{

if(b==0)

throw "value of b is zero**\n**";

else

throw "value of b is less than zero**\n**";

}

}

catch(const char\* e)

{

cout<<e;

}

}

a) value of b is zero  
b) value of b is less than zero  
c) Product of 5 and -1 is: -5  
d) Aborted(core dumped)

Answer: b  
Explanation: Here the func() throws the value of b which is caught by the inner try-catch block, which again throws the message inorder to handle different cases of b which is caught by the outer try-catch block. Now as the value of b is negative the program outputs the message as shown.  
Output:

$ ./a.out

value of b is less than zero

# C++ Programming Questions and Answers – Exception Handling – 2

1. Where should we place catch block of the derived class in a try-catch block?  
a) Before the catch block of Base class  
b) After the catch block of Base class  
c) Anywhere in the sequence of catch blocks  
d) After all the catch blocks

Answer: a  
Explanation: C++ asks the programmer to place the catch block of derived class before a catch block of the base class, otherwise derived catch block will never be executed.

2. What happens when this C++ program is compiled?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

class A

{

int a;

public:

A(){}

};

class B: public A

{

int b;

public:

B(){}

};

void func()

{

B b;

throw b;

}

int main()

{

try{

func();

}

catch(A a){

cout<<"Caught A Class**\n**";

}

catch(B b){

cout<<"Caught B Class**\n**";

}

}

a) The program compiles successfully without any errors or warnings  
b) Compile-time error occurs  
c) The program compiles successfully with warnings  
d) The program gives both errors and warnings

Answer: c  
Explanation: Catch block of derived should always be placed before the catch block base class, hence the program gives warnings stating that exceptions of the derived class will be caught by the base class.  
Output:

$ g++ check.cpp

check.cpp: In function ‘int main()’:

check.cpp:33:2: warning: exception of type ‘B’ will be caught

catch(B b){

^~~~~

check.cpp:30:2: warning: by earlier handler for ‘A’

catch(A a){

^~~~~

3. What will be the output of the following C++ code?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

class A

{

int a;

public:

A(){}

};

class B: public A

{

int b;

public:

B(){}

};

void func()

{

B b;

throw b;

}

int main()

{

try{

func();

}

catch(A a){

cout<<"Caught A Class**\n**";

}

catch(B b){

cout<<"Caught B Class**\n**";

}

}

a) Caught B Class  
b) Caught A Class  
c) Compile-time error  
d) Run-time error

Answer: b  
Explanation: As the catch block of the derived class is after the catch block of base class, therefore, all the exceptions of the derived class will be caught by the base class, Hence the output of catch block of class A is printed.  
Output:

$ ./a.out

Caught A Class

4. What will be the output of the following C++ code?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

class A

{

int a;

public:

A(){}

};

class B: public A

{

int b;

public:

B(){}

};

void func()

{

B b;

throw b;

}

int main()

{

try{

func();

}

catch(B b){

cout<<"Caught B Class**\n**";

}

catch(A a){

cout<<"Caught A Class**\n**";

}

}

a) Caught B Class  
b) Caught A Class  
c) Compile-time error  
d) Run-time error

Answer: a  
Explanation: In this as the catch block of the derived class is before the catch block of the base class so when func() throws the object of class B it is caught by the catch block of class B, Hence the output is printed as shown.  
Output:

$ ./a.out

Caught B Class

5. What will be the output of the following C++ code?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

class A

{

int a;

public:

A(){}

};

class B: public A

{

int b;

public:

B(){}

};

void func()

{

B b;

throw b;

}

int main()

{

try{

func();

}

catch(B \*b){

cout<<"Caught B Class**\n**";

}

catch(A a){

cout<<"Caught A Class**\n**";

}

}

a) Caught B Class  
b) Caught A Class  
c) Compile-time error  
d) Run-time error

Answer: b  
Explanation: The func() throws the object of class B but as catch block is defined to catch the exception of class B, Therefore the exception is caught by the base class A. The programmer has defined the catch block for B\*, therefore, the object B is not caught by the pointer object B\*.

6. What id the syntax for catching any type of exceptions?  
a) catch(Exception e)  
b) catch(…)  
c) catch(Exception ALL)  
d) catch(ALL)

Answer: b  
Explanation: catch(…) is used in C++ to catch all types of exceptions in a single catch block.

7. What will be the output of the following C++ code?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

class A

{

int a;

public:

A(){}

};

class B: public A

{

int b;

public:

B(){}

};

void func1()

{

B b;

throw b;

}

void func2()

{

A a;

throw a;

}

int main()

{

try{

func1();

}

catch(...){

cout<<"Caught All types of exceptions**\n**";

}

try{

func2();

}

catch(B b){

cout<<"Caught All types of exceptions**\n**";

}

}

a) Caught All types of exceptions  
b)Caught All types of exceptions

Aborted (core dumped)

c) Error  
d)Caught All types of exceptions

Caught All types of exceptions

Answer: b  
Explanation: Two try-catch blocks is declared each catching the respective exceptions from class A and B. But as we have defined catch all exceptions in the first case, therefore, the exception for class B is caught when thrown by the func1(), but in the second case, the try-catch block is catching only the exception for class B so when func2() throws class A exception and no catch block to catch that exception therefore program results into abort(core dumped).

8. Uncaught exception leads to \_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) termination of program  
b) successful execution of programs  
c) no effect on the program  
d) execution of other functions of the program starts

Answer: a  
Explanation: Uncaught exceptions in a program leads to the termination of a program.

9. An uncaught handler returns to \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) main function  
b) its caller  
c) its callee  
d) none of the mentioned

Answer: d  
Explanation: Uncaught exceptions do not “return” to any specific location in the program. They trigger a chain of events leading to program termination. In C++, when an uncaught exception occurs, the program unwinds the stack to find an appropriate exception handler. It searches backward through the call stack (i.e., the series of function calls that led to the point where the exception occurred) until it finds a function that has a suitable exception handler. If no handler is found, the program may terminate or exhibit undefined behavior.

10. Header file used for exception handling in C++?  
a) <cstdlib>  
b) <string>  
c) <handler>  
d) <exception>

Answer: d  
Explanation: <exception> header file is used to use exception handler in C++

# C++ Programming Questions and Answers – Exception Handling – 3

1. The C++ code which causes abnormal termination/behaviour of a program should be written under \_\_\_\_\_\_\_\_\_ block.  
a) try  
b) catch  
c) finally  
d) throw

Answer: a  
Explanation: Code that leads to the abnormal termination of the program should be written under the try block.

2. Exception handlers are declared with \_\_\_\_\_\_\_\_\_\_\_\_ keyword.  
a) try  
b) catch  
c) throw  
d) finally

Answer: b  
Explanation: C++ uses catch block to handle any exceptions that occur during run-time of the program.

3. Which of the following statements are correct about Catch handler?

i. It must be placed immediately after the try block

ii. It can have more than one parameters

iii. There must be one and only one catch handler for every try block

iv. There can be multiple catch handler for a try block

v. General catch handler can be kept anywhere after try block.

a) i, iv, v  
b) i, ii, iii  
c) i, iv  
d) i, ii

Answer: c  
Explanation: A catch block should always be placed after the try block and there can be multiple catch block following a try block.

4. In nested try-catch block, if the inner catch block gets executed, then\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) Program stops immediately  
b) Outer catch block also executes  
c) Compiler jumps to the outer catch block and executes remaining statements of the main() function  
d) Compiler executes remaining statements of outer try-catch block and then the main() function

Answer: d  
Explanation: The inner catch block will be executed then remaining part of the outer try block will be executed and then the main bock will be executed.

5. If inner catch block is unable to handle the exception thrown then\_\_\_\_\_\_\_\_\_\_  
a) The compiler looks for the outer try-catch block  
b) Program stops abnormally  
c) The compiler will check for appropriate catch handler of the outer try block  
d) The compiler will not check for appropriate catch handler of the outer try block

Answer: c  
Explanation: In such cases, the compiler will try to find an appropriate outer catch block to handle the exception otherwise if nothing is there then occurs the abnormal behaviour of the program.

6. In nested try catch blocks, if both inner and outer catch blocks are unable to handle the exception thrown, then \_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) Compiler executes only main()  
b) Compiler throws compile time errors about it  
c) Program will run without any interrupt  
d) Program will be termianted abnormally

Answer: d  
Explanation: If no inner/outer catch handler is avaliable to handle the exception then as usual the program will show abnormal behaviour.

7. Which function is invoked when an unhandled exception is thrown?  
a) stop()  
b) aborted()  
c) terminate()  
d) abandon()

Answer: c  
Explanation: terminate() function is called/invoked incase any exception is not handled properly.

8. How one can restrict a function to throw particular exceptions only?  
a) By defining multiple try-catch blocks inside a function  
b) By defining a generic function within a try-catch block  
c) By defining a function with throw clauses  
d) Not allowed in C++

Answer: c  
Explanation: We can use throw clause to mention the exceptions that a function can throw. Hence restricting the function to throw some particular exceptions only.

9. Which function is invoked when we try to throw an exception that is not supported by a function?  
a) indeterminate()  
b) unutilized()  
c) unexpected()  
d) unpredicted()

Answer: c  
Explanation: As the exception is not supported by the function so it does not know what to do about the exception in that case it call the unexpected() function of the STL library.

10. Return type of uncaught\_exception() is\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
a) int  
b) bool  
c) char \*  
d) double

Answer: b  
Explanation: Return type of uncaught exceptions are bool.

11. What will be the output of the following C++ code?

#include <iostream>

using namespace std;

int main()

{

int var = -12;

try {

cout<<"Inside try**\n**";

if (var < 0)

{

throw var;

cout<<"After throw**\n**";

}

}

catch (int var ) {

cout<<"Exception Caught**\n**";

}

cout<<"After catch**\n**";

return 0;

}

a)Inside try

Exception Caught

After catch

b)Inside try

After throw

After catch

c)Inside try

Exception Caught

After throw

d)Inside try

Exception Caught

After throw

After catch

Answer: a  
Explanation: “Inside try” will always be printed as we just entering try block then. Now as var < 0 therefore the try block will throw int var as exception hence “After throw” will not be printed) Now this exception will be caught by the catch handler printing “Exception caught” and at last after terminating the program “After catch” will be printed.

11. What will be the output of the following C++ code?

#include <iostream>

using namespace std;

int main()

{

int var = -12;

try {

cout<<"Inside try**\n**";

if (var < 0)

{

throw var;

cout<<"After throw**\n**";

}

}

catch (char var ) {

cout<<"Exception Caught**\n**";

}

cout<<"After catch**\n**";

return 0;

}

a)Inside try

Exception Caught

After catch

b)Inside try

After throw

After catch

c) Error  
d) Run-time error

Answer: d  
Explanation: As no catch handler is defined to catch an integer hence when var variable, which is int, is thrown then nothing is there to catch the int hence the program terminates abnormally.

12. What will be the output of the following C++ code?

#include <iostream>

using namespace std;

int main()

{

try

{

try

{

throw 20;

}

catch (int n)

{

cout << "Inner Catch**\n**";

}

}

catch (int x)

{

cout << "Outer Catch**\n**";

}

return 0;

}

a) Inner Catch  
b) Outer Catch  
c)Inner Catch

Outer Catch

d) Error

Answer: a  
Explanation: As exception thrown by the inner try block is caught by the inner catch block hence the exception is handled at the inner level and program continues to run outer try block, statement afterwards.

13. What will be the output of the following C++ code?

#include <iostream>

using namespace std;

int main()

{

try

{

try

{

throw 20;

}

catch (char n)

{

cout << "Inner Catch**\n**";

}

}

catch (int x)

{

cout << "Outer Catch**\n**";

}

return 0;

}

a) Inner Catch  
b) Outer Catch  
c)Inner Catch

Outer Catch

d) Error

Answer: b  
Explanation: As there is no inner catch handler to handle the int exception thrown by the try block therefore outer catch block handler catches the exception thrown by the inner try catch therefore the output prints “Outer Catch” instead of “Inner Catch”. After that program continues execution.

14. What will be the output of the following C++ code?

#include <iostream>

using namespace std;

int main()

{

try

{

try

{

throw 20;

}

catch (int n)

{

cout << "Inner Catch**\n**";

throw;

}

}

catch (int x)

{

cout << "Outer Catch**\n**";

}

return 0;

}

a) Inner Catch  
b) Outer Catch  
c)Inner Catch

Outer Catch

d) Error

Answer: c  
Explanation: The exception thrown by the inner try catch block is caught by the inner block hence “Inner Catch” is printed but as inner catch block again throws an exception further therefore the exception is thrown further which is caught by the outer catch block hence “Outer Catch” is also printed.

15. Which of the following is true about exception handling in C++?

i) There is a standard exception class in C++ similar to Exception class in Java.

ii) All exceptions are unchecked in C++, i.e., the compiler does not checks if the exceptions are caught or not.

iii) In C++, a function can specify the list of exceptions that it can throw using comma separated list like following.

void fun(int a, char b) throw (Exception1, Exception2, ..)

a) i, iii  
b) i, ii, iii  
c) i, ii  
d) ii, iii

Answer: b  
Explanation: In C++ also we have an exception class similar to java. All exceptions are unchecked in C++. We can specify the list of exception that a function throws using the above format.

# C++ Programming Questions and Answers – Error Handling Alternatives

1. Which alternative can replace the throw statement?  
a) for  
b) break  
c) return  
d) exit

Answer: c  
Explanation: throw and return does the same job as return a value. So it can be replaced.

2. What are the disadvantages if use return keyword to return error codes?  
a) You have to handle all exceptional cases explicitly  
b) Your code size increases dramatically  
c) The code becomes more difficult to read  
d) All of the mentioned

Answer: d  
Explanation: As we are using return for each and every exception, It will definitely increase the code size.

3. What is most suitable for returning the logical errors in the program?  
a) Use constructor and destructor  
b) Set a global error indicator  
c) Use break keyword  
d) Use final keyword

Answer: b  
Explanation: Set a global error indicator is most suitable for returning the logical errors in the program.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <typeinfo>
3. using namespace std;
4. class A
5. {
6. };
7. int main()
8. {
9. char c; float x;
10. if (typeid(c) != typeid(x))
11. cout << typeid(c).name() << endl;
12. cout << typeid(A).name();
13. return 0;
14. }

a)c

1A

b) x  
c) Both c & x  
d) c

Answer: a  
Explanation: We are checking the type id of char and float as they are not equal, We are printing c.  
Output:

$ g++ eal.cpp

$ a.out

c

1A

5. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. void Division(const double a, const double b);
4. int main()
5. {
6. double op1=0, op2=10;
7. try
8. {
9. Division(op1, op2);
10. }
11. catch (const char\* Str)
12. {
13. cout << "**\n**Bad Operator: " << Str;
14. }
15. return 0;
16. }
17. void Division(const double a, const double b)
18. {
19. double res;
20. if (b == 0)
21. throw "Division by zero not allowed";
22. res = a / b;
23. cout << res;
24. }

a) 0  
b) Bad operator  
c) 10  
d) 15

Answer: a  
Explanation: We are dividing 0 and 10 in this program and we are using the throw statement in the function block.  
Output:

$ g++ eal.cpp

$ a.out

0

6. What will be the output of the following C++ code?

1. #include <stdexcept>
2. #include <limits>
3. #include <iostream>
4. using namespace std;
5. void MyFunc(char c)
6. {
7. if (c < numeric\_limits<char>::max())
8. return invalid\_argument;
9. }
10. int main()
11. {
12. try
13. {
14. MyFunc(256);
15. }
16. catch(invalid\_argument& e)
17. {
18. cerr << e.what() << endl;
19. return -1;
20. }
21. return 0;
22. }

a) 256  
b) Invalid argument  
c) Error  
d) 246

Answer: c  
Explanation: We can’t return a statement by using the return keyword, So it is arising an error.

7. What is the use of RAII in c++ programming?  
a) Improve the exception safety  
b) Terminate the program  
c) Exit from the block  
d) Crash the compiler

Answer: a  
Explanation: RAII is used to improve the exception safety.

8. How many levels are there in exception safety?  
a) 1  
b) 2  
c) 3  
d) 4

Answer: c  
Explanation: The three levels of exception safety are basic, strong and no throw.

9. Pick out the correct statement for error handling alternatives.  
a) Terminate the program  
b) Use the stack  
c) Exit from the block  
d) Use the queue

Answer: b  
Explanation: When an error is raised means, it will be pushed into stack and it can be corrected later by the programmer.

10. What will happen when an exception is not processed?  
a) It will eat up a lot of memory and program size  
b) Terminate the program  
c) Crash the compiler  
d) Displays proper output

Answer: a  
Explanation: As in the case of not using an exception, it will remain useless in the program and increase the code complexity.

# C++ Programming Questions and Answers – Standard Exceptions

1. Which header file is used to declare the standard exception?  
a) #include<exception>  
b) #include<except>  
c) #include<error>  
d) #include<exce>

Answer: a  
Explanation: #include<exception> is used to declare the standard exception.

2. Where are standard exception classes grouped?  
a) namespace std  
b) error  
c) catch  
d) final

Answer: a  
Explanation: As these are standard exceptions, they need to be defined in the standard block, So it is defined under namespace std.

3. How many types of standard exception are there in c++?  
a) 9  
b) 5  
c) 6  
d) 7

Answer: a  
Explanation: There are nine standard exceptions in c++. They are bad\_alloc, bad\_cast, bad\_exception, bad\_function\_call, bad\_typeid, bad\_weak\_ptr, ios\_base::failure, logic\_error and runtime\_error.

4. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. class myexc: public exception
5. {
6. virtual const char\* what() const throw()
7. {
8. return "My exception";
9. }
10. } myex;
11. int main ()
12. {
13. try
14. {
15. throw myex;
16. }
17. catch (exception& e)
18. {
19. cout << e.what() << endl;
20. }
21. return 0;
22. }

a) My  
b) My exception  
c) No exception  
d) exception

Answer: b  
Explanation: This is a type of exception arising in the class. We can call this  
also as a standard exception.  
Output:

$ g++ std.cpp

$ a.out

My exception

5. What will be the output of the following C++ code?

1. #include <iostream>
2. #include <exception>
3. using namespace std;
4. int main ()
5. {
6. try
7. {
8. int\* myarray= new int[1000];
9. cout << "Allocated";
10. }
11. catch (exception& e)
12. {
13. cout << "Standard exception: " << e.what() << endl;
14. }
15. return 0;
16. }

a) Allocated  
b) Standard exception:  
c) bad\_alloc  
d) Depends on memory

Answer: d  
Explanation: Variable will be allocated depends on the available space in the memory, If there is no space means, It will throw an exception.  
Output:

$ g++ std1.cpp

$ a.out

Allocated

6. What will be the output of the following C++ code?

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. char\* ptr;
6. unsigned long int a = (size\_t(0) / 3);
7. cout << a << endl;
8. try
9. {
10. ptr = new char[size\_t(0) / 3];
11. delete[ ] ptr;
12. }
13. catch(bad\_alloc &thebadallocation)
14. {
15. cout << thebadallocation.what() << endl;
16. };
17. return 0;
18. }

a) 0  
b) 2  
c) bad\_alloc  
d) depends on compiler

Answer: a  
Explanation: As we are dividing the zero by three, it is returning 0.  
Output:

$ g++ std2.cpp

$ a.out

0

7. What will be the output of the following C++ code?

1. #include <typeinfo>
2. #include <iostream>
3. using namespace std;
4. class shape
5. {
6. public:
7. virtual void myvirtualfunc() const {}
8. };
9. class mytriangle: public shape
10. {
11. public:
12. virtual void myvirtualfunc() const
13. {
14. };
15. };
16. int main()
17. {
18. shape shape\_instance;
19. shape &ref\_shape = shape\_instance;
20. try
21. {
22. mytriangle &ref\_mytriangle = dynamic\_cast<mytriangle&>(ref\_shape);
23. }
24. catch (bad\_cast)
25. {
26. cout << "Caught: bad\_cast exception**\n**";
27. }
28. return 0;
29. }

a) Caught standard exception  
b) No exception arises  
c) Caught: bad\_cast exception  
d) Caught: cast

Answer: c  
Explanation: As we are not able to allocate the values by using dynamic cast,  
So it is arising an exception.  
Output:

$ g++ std3.cpp

$ a.out

Caught: bad\_cast exception

8. What will be the output of the following C++ code?

1. #include <typeinfo>
2. #include <iostream>
3. using namespace std;
4. class Test
5. {
6. public:
7. Test();
8. virtual ~Test();
9. };
10. int main()
11. {
12. Test \*ptrvar = NULL;
13. try
14. {
15. cout << typeid(\*ptrvar).name() << endl;
16. }
17. catch (bad\_typeid)
18. {
19. cout << "The object is null" << endl;
20. }
21. return 0;
22. }

a) No exception arises  
b) The object is null  
c) Error  
d) The object is

Answer: b  
Explanation: As there is no object in the class, It is arising an exception in the program.  
Output:

$ g++ std4.cpp

$ a.out

The object is null

9. Which of the following is best to include under try block?  
a) static values  
b) const values  
c) dynamic allocations  
d) default values

Answer: c  
Explanation: Because the dynamic allocations can change at any time, So it is best to include in try block.

10. What are the predefined exceptions in c++?  
a) Memory allocation errors  
b) I/O errors  
c) Both Memory allocation errors & I/O errors  
d) static errors

Answer: c  
Explanation: Both Memory allocation errors & I/O errors are the predefined exceptions in c++.